# Intro: Url Endpoints used

App menu tree:

Main menu - https://mylinks.aghayev.com/api

Sub menu: https://mylinks.aghayev.com/api/subcategories

Menu page: https://mylinks.aghayev.com/api/weblinks/{slug}

# Intro: App folder structure

App router:

app/

Navbar.js – ISR, fetch { revalidate: 10 }

weblinks/[slug]/page.tsx –

SSG – generateStaticParams

protected/*

services/*

lib/*

# Navbar.js – ISR

ISR – Incremental Static Regeneration.

Used if page content depends on external data

App Router:

```
const res = await fetch('https://.../posts',{ next: { revalidate: 10 } })
```

Pages Router:

```
export async function getStaticProps() {
  const res = await fetch('https://.../posts')
  const posts = await res.json()
  return { props: { posts }, revalidate: 10 }
}
```

npm run build

Result: If client-side data fetch - static site data stale for 10 seconds, then re-fetches.

If SSR and revalidate used - data re-fetched every 10 seconds

# weblinks/[slug]/page.tsx - SSG

SSG – Static Site Generation

Used for statically generate routes

App router, app/weblinks/[slug]/page.tsx:

```
export async function generateStaticParams() {
 const res = await fetch('https://.../subcategories')
  const categories = await res.json()
 return categories.map(({ slug }) => ({
    slug: slug,
  }))}
```

Pages router:

getStaticPaths()

npm run build

```
└ ● /weblinks/[slug]                          182 B          87.4 kB
    ├ /weblinks/general
    ├ /weblinks/finance
    ├ /weblinks/my-livejournal
    └ [+13 more paths]

●  (SSG)     prerendered as static HTML (uses getStaticProps)
```

# App Router file convension loading.js

app/weblinks/[slug]/loading.tsx

```
export default function Loading() {
  return <p>Loading...</p>
}
```

# Capacitor Installation, page 1

package.json

"static": "next build && next export"

next.config.mjs

const nextConfig = { output:‘export’}

export default nextConfig

capacitor.config.js

…"webDir": "out"

# Capacitor installation, page 2

Installation:

npm i @capacitor/core

npm i -D @capacitor/cli

npx cap init

npm i @capacitor/android @capacitor/ios

Build time:

npm run build

npx cap add ios (npx cap add android)

npx cap sync ( next.js codebase translation to ios/ or android/ codebase) – if fails for ios/ - see Capacitor Installation 3

IOS: Code copied to: /ios/App/App/public

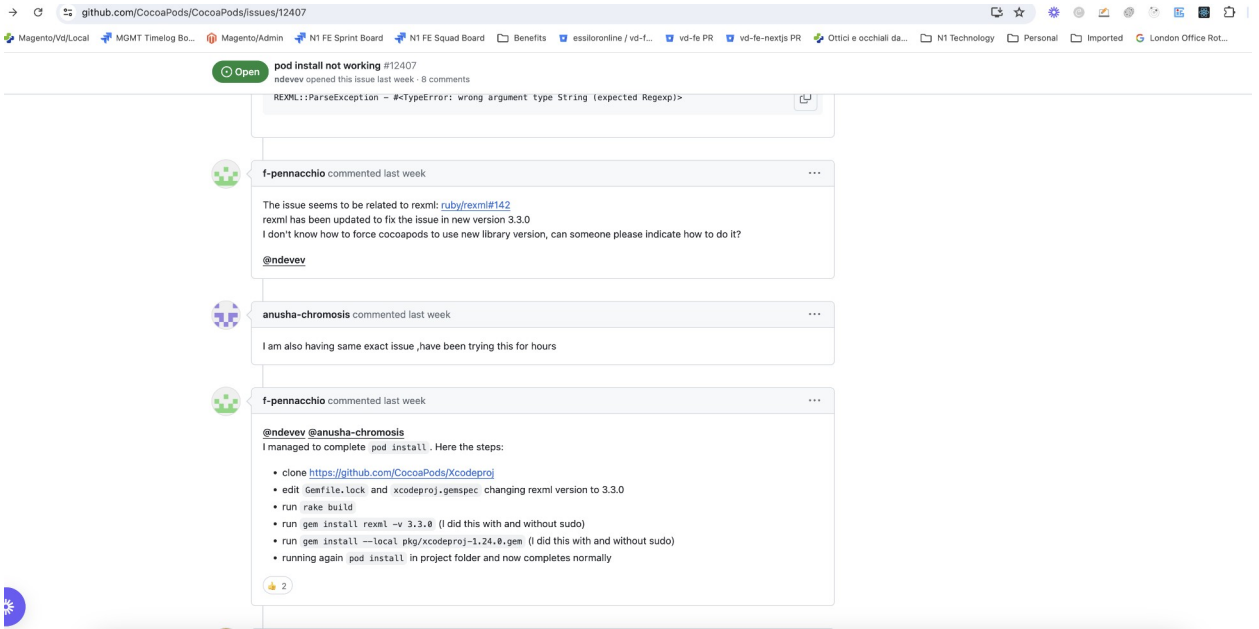
Android: Code copied to: /android/app/src/main/assets/public

npx cap open ios (or npx cap open android – opens Xcode IDE or Android Studio)

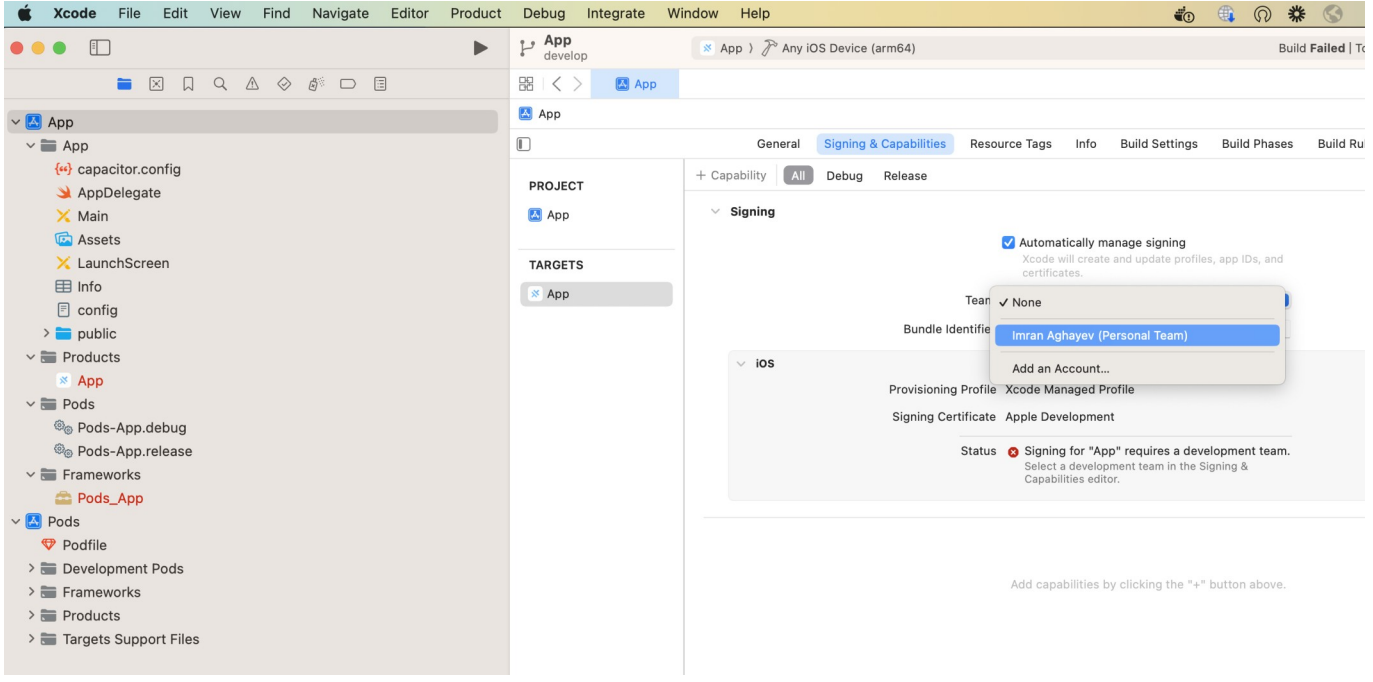
# Capacitor Installation, page 3

For ios/ on some Macs build fails

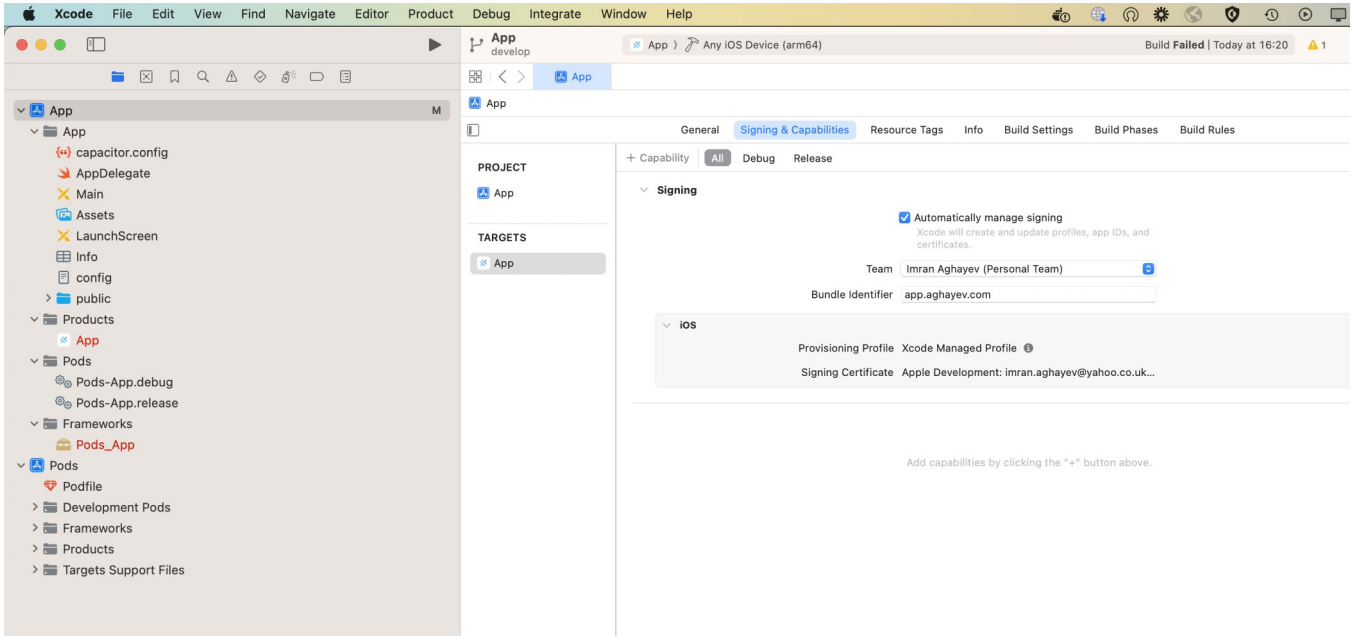
To fix: follow this: [Xcodeproj CocoaPods]

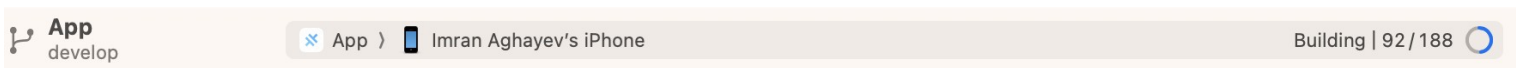


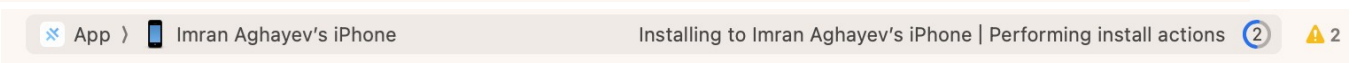
# Deploying to IOS, page 1

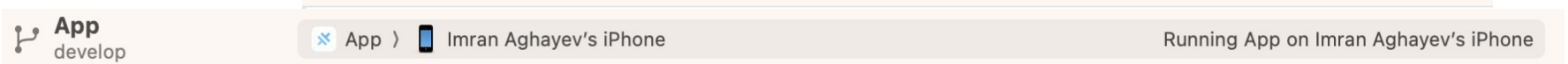



# Deploying to IOS, page 2

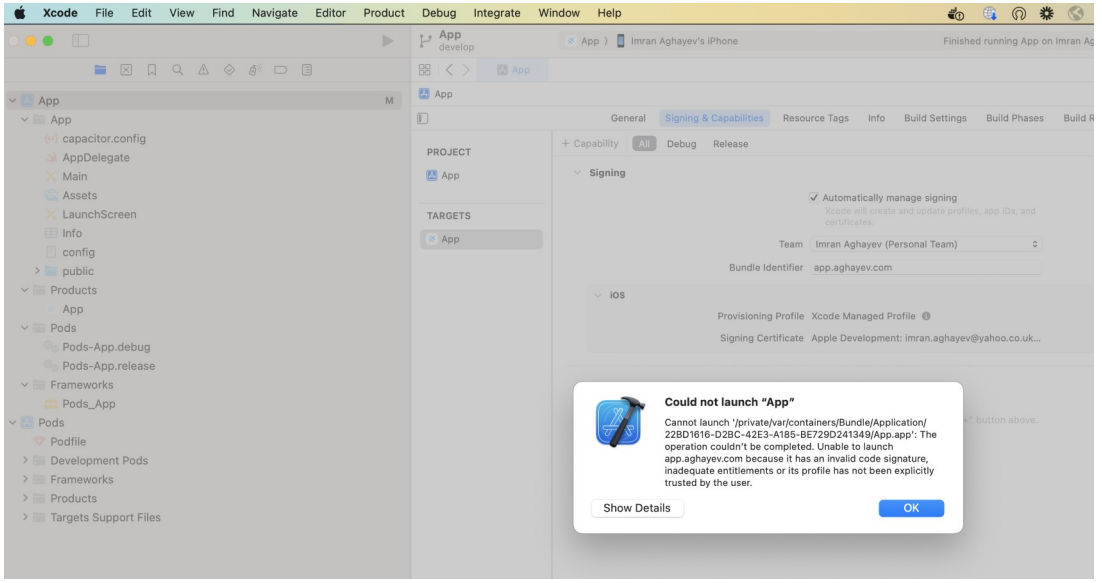


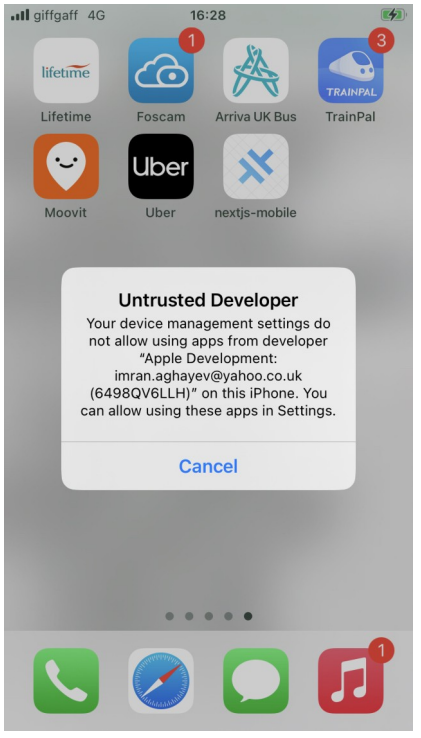



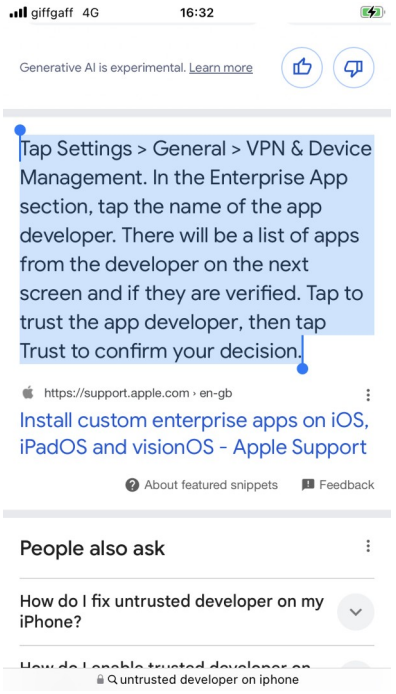



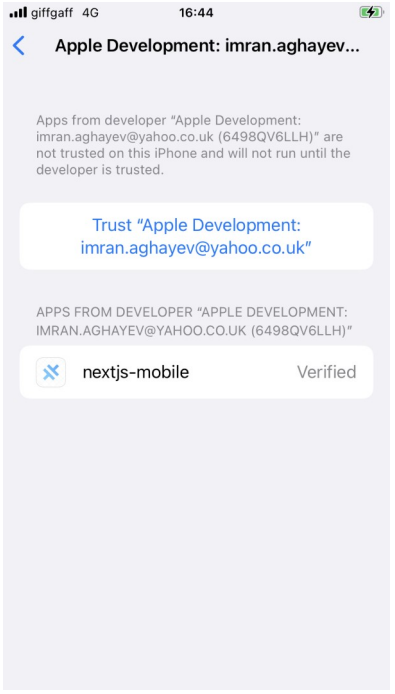
# Deploying to IOS, page 3

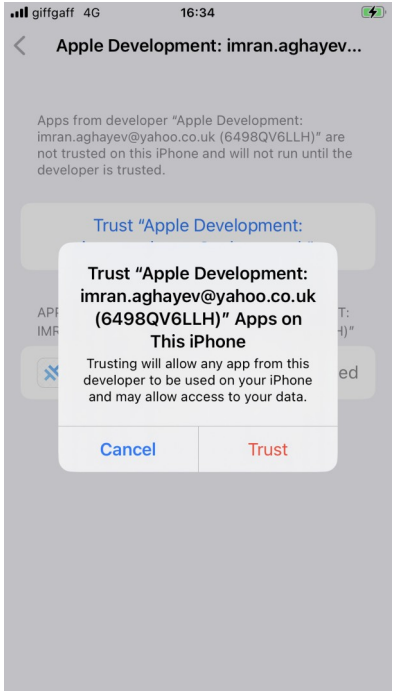


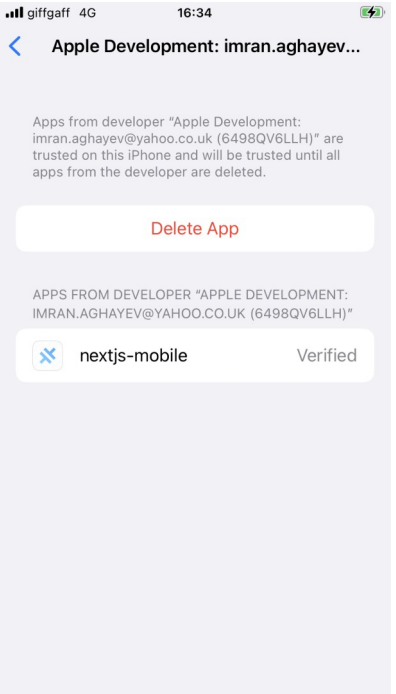



# Deploying to IOS, page 4



# Resources:

1. Next.Js Mobile App in minutes with Capacitor
2. Capacitor Installation Instruction
3. Capacitor official documentation
4. Adding Signing Capabilities to your IOS App
5. Deploying to Android phone
5. Next.Js SSG and ISR
6. Next.Js App Router Migration